

CCCCCCCCCCCCC		JJJ	FFFFFFFFFFFFFF	VVV	VVV	444	444
CCCCCCCCCCCCC		JJJ	FFFFFFFFFFFFFF	VVV	VVV	444	444
CCCCCCCCCCCCC		JJJ	FFFFFFFFFFFFFF	VVV	VVV	444	444
CCC		JJJ	FFF	VVV	VVV	444	444
CCC		JJJ	FFF	VVV	VVV	444	444
CCC		JJJ	FFF	VVV	VVV	444	444
CCC		JJJ	FFF	VVV	VVV	444	444
CCC		JJJ	FFF	VVV	VVV	444	444
CCC		JJJ	FFF	VVV	VVV	444	444
CCC		JJJ	FFF	VVV	VVV	444	444
CCC		JJJ	FFFFFFFFFFFFFF	VVV	VVV	4444444444444444	
CCC		JJJ	FFFFFFFFFFFFFF	VVV	VVV	4444444444444444	
CCC		JJJ	FFFFFFFFFFFFFF	VVV	VVV	4444444444444444	
CCC	JJJ	JJJ	FFF	VVV	VVV		444
CCC	JJJ	JJJ	FFF	VVV	VVV		444
CCC	JJJ	JJJ	FFF	VVV	VVV		444
CCC	JJJ	JJJ	FFF	VVV	VVV		444
CCC	JJJ	JJJ	FFF	VVV	VVV		444
CCC	JJJ	JJJ	FFF	VVV	VVV		444
CCC	JJJ	JJJ	FFF	VVV	VVV		444
CCCCCCCCCCCCC	JJJJJJJJJ	JJJ	FFF	VVV	VVV		444
CCCCCCCCCCCCC	JJJJJJJJJ	JJJ	FFF	VVV	VVV		444
CCCCCCCCCCCCC	JJJJJJJJJ	JJJ	FFF	VVV	VVV		444

[illegible]

{ \$begin JNLDEFINT.V04-000

{* *****
{* COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
{* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
{* ALL RIGHTS RESERVED.
{* *****

{* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
{* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
{* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
{* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
{* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
{* TRANSFERRED.
{* *****

{* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
{* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
{* CORPORATION.
{* *****

{* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
{* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
{* *****

{++ Facility: JOURNALING : DEFINITION OF INTERNAL SYMBOLICS

{ Abstract:

{ This module contains the symbolic definitions for non-user accessible
{ data structures.

{ Author: Joost Verhofstad

{ Modified by:

{ V03-052 MKL0210 Mary Kay Lyons 16-DEC-1983
{ Add MCSID field to JNLMSG UCBDATA message.
{ V03-051 MKL0199 Mary Kay Lyons 29-NOV-1983
{ Add JNLMSG\$W_JNL_PROT.
{ V03-050 LY0418 Larry Yetto 27-SEP-1983 15:10:06
{ Add EPID and ARB_PRIV to JNLCWQDEF
{ V03-049 LY0415 Larry Yetto 13-SEP-1983 10:40:11
{ Add REQCSB and some spare fields to JNLBTX structure
{ V03-048 MKL0168 Mary Kay Lyons 23-AUG-1983
{ Add STS field to JNLMSG UCBDATA message.
{ V03-047 LY0406 Larry Yetto 3-AUG-1983 08:57:57
{ Fix JNLMSGDATA structure. Change IOSTS\$V_RESUBS
{ to IOSTS\$V_RESUB

V03-046 LY0405 Larry Yetto 2-AUG-1983 14:45:47
Add JNLMSGDATA structure

V03-045 LY0403 Larry Yetto 1-AUG-1983 15:18:18
Add JNLBXSTSSV_FNCTCMPL and JNLBXSTSSV_CNXBK

V03-044 LY0399 Larry Yetto 28-JUL-1983 15:37:31
Add JNLBXSTS and JNLBTX structures to hold information relevant
to block transfer operations in progress that were initiated
from some other node.

V03-043 MKL0132 Mary Kay Lyons 24-JUL-1983
Change JNLRC to contain an offset to filter information.

V03-042 MKL0126 Mary Kay Lyons 10-JUL-1983
Remove JNLRSB_JNLTP definition. Add JNLMSG
definitions for creating journaling I/O database.
Define JNLRC\$Q_DATTIM to overlay JNLRC\$Q_RUID.
Keep the file version number in the JMT. Remove
IOSTSSM_REM_WRITE and IOSTSSV_REM_WRITE.
Make journal names 12 bytes and various changes for send-
journal-message stuff.

V03-041 MKL0116 Mary Kay Lyons 22-JUN-1983
Add pointer to mount item list in the ADB. Add
UPDATE_ADL message definitions.

V03-040 LY0383 Larry Yetto 16-JUN-1983 17:43:21
Move cluster message dispatch codes to [SYSLOA.SRC]CLUSTER.SDL

V03-039 PRB0196 Paul Beck 12-JUN-1983 14:20
Add RUE\$V_NOFAC, RUE\$V_NOOBJ.

V03-038 MKL0096 Mary Kay Lyons 01-Jun-1983
Add JNLRCDEF.

V03-037 MKL0093 Mary Kay Lyons 27-MAY-1983
Replace missing JNLMSGDEF.

V03-036 LY0373 Larry Yetto 24-MAY-1983 15:52:40
Add new BCB fields for high sequence number completely
in the buffer and written. Add JNLWCQ structure. Add
fields to overlay RUE\$Q_RUID.

V03-035 MKL0087 Mary Kay Lyons 19-MAY-1983
Change JNLMSGDEF.

V03-034 JSV0289 Joost Verhofstad 18-MAY-1983
Reorganize and split up into:
JNLDEFINT.SDL
JNLSYSDEF.SDL
JNLACPDEF.SDL
JNLFILE.SDL

V03-033 LY0361 Larry Yetto 9-MAY-1983 12:32:19
Rename CJLMSG macro to CJFMSGFNC. Add JNLACBM.

Add JNLLOGSV_SLVCRFAIL. Remove JNLCB def.

V03-032	JSV0229	Joost Verhofstad	27-APR-1983
	Add RUSYNC bits		
V03-031	LY0355	Larry Yetto	20-APR-1983 10:03:27
	Add cluster message dispatch codes CJLMSG macro and remove the obsolete SCS message crap.		
	Remove ENT_TYPE codes and bit definitions from FLTR macro		
V03-030	MKL0068	Mary Kay Lyons	08-APR-1983
	Add RCB\$\$_LSTBLK1 and RCB\$\$_LSTBLK2.		
V03-029	JSV0212	Joost Verhofstad	06-APR-1983
	Change ACP filter to contain two date-time fields		
V03-028	LY0346	Larry Yetto	6-APR-1983 11:03:17
	Add the JNLCB structure. This structure is the Journal control block for slave nodes with no channels.		
V03-027	MKL0062	Mary Kay Lyons	30-MAR-1983
	Add JFTE\$\$_FRSTJVB, RCB\$\$_LSTBLK1, and RCB\$\$_LSTBLK2.		
V03-026	JSV190	Joost Verhofstad	14-MAR-1983
	Add JFTE fields		
V03-025	MKL0048	Mary Kay Lyons	24-FEB-1983
	Update comments for JFTE\$\$_JMT and JFTE\$\$_DEVNAM.		
V03-024	JSV0151	Joost Verhofstad	17-FEB-1983
	Add JMT\$\$_BASEVBN and JMT\$\$_LTVBN and SFT\$\$_BASEVBN		
V03-023	JSV0144	Joost Verhofstad	14-FEB-1983
	Add BCB\$\$_NWVPR		
V03-022	JSV0141	Joost Verhofstad	09-FEB-1983
	Add JFTE\$\$_NEXTVER		
V03-021	JSV0137	Joost Verhofstad	03-FEB-1983
	replace source, put in null packet		
V03-020	LY0245	Larry Yetto	10-JAN-1983
	Move RUS structure to JNLDEF.SDL		
V03-19	JSV0116	Joost Verhofstad	04-Jan-1983
	Remove PROCNAME, BINARY, PROCNODE, PROCGROUP, PROCRUNTIME fields from FLTR structure		
V03-18	JSV0107	Joost Verhofstad	04-Jan-1983
	Fix RCB fields + commentary		
V03-17	JSV0106	Joost Verhofstad	30-Dec-1982
	Add RCB fields		
V03-16	JSV0105	Joost Verhofstad	12-Dec-1982
	Add JFTE field		

V03-15 JSV0097 Joost Verhofstad 23-Nov-1982
Add OPCHDR data structure

V03-14 JSV0087 Joost Verhofstad 28-Oct-1982
Add TBUF data structure and GTB, RCB, RHD
and JFTE symbols.

V03-13 JSV0078 Joost Verhofstad 08-Oct-1982
Add CJL data structure

V03-12 JSV0064 Joost Verhofstad 22-Sep-1982
Add a few GTB, JMT, JFTE fields for tape reading

V03-011 JSV0054 Joost Verhofstad 26-Aug-1982
Add FLTR\$V_OUTRANGE and FLTR\$\$_OUTRANGE

V03-010 JAY0007 John A. Ywoskus 02-Aug-1982
Generate \$M's for status bits in RUE and RUS.

V03-009 JAY0006 John A. Ywoskus 21-Jul-1982
Add INDEX field to RUS.

V03-008 JSV0024 Joost Verhofstad 21-Jul-1982
Add JNLLOG bits

V03-007 JAY0005 John A. Ywoskus 21-Jul-1982
Make RUE\$W_JNLCNT be a longword. Add this field to RUS.

V03-006 JAY0004 John A. Ywoskus 15-Jul-1982
Change RUS structure. Delete WRFLG and add entry
attributes. Add COUNT field to NDL.

V03-005 JAY0003 John A. Ywoskus 12-Jul-1982
Add JNLCNT field to RUE.

V03-004 JSV000 Joost Verhofstad 7-Jul-1982
BUFFER\$W_JNLID => BUFFER\$L_JNLID

V03-003 JAY0002 John A. Ywoskus 06-Jul-1982
Rename RULIST structure to RUS. Change 'RESIDUAL'
status to RESID_FOR and RESID_BCK in RUE and RUS.
Add an 'INDEX' field to RUE.

V03-002 LY0028 Larry Yetto 29-Jun-1982
Added Name table Device List (NDL) definition

V03-001 JAY0001 John A. Ywoskus 17-Jun-1982
Added JNLDB, message structures for cluster journaling.
Delete RUDEF structure, replace with a version of RULIST.


```
module $CJFFLGDEF;
```

```
/*++
```

```
/*
```

```
/* CJFFLG - Flags that can be returned from SENSEMODE
```

```
/*
```

```
/*--
```

```
aggregate CJFFLGDEF union fill prefix CJFFLG$;
```

```
  CJFFLGDEF BITS structure fill;
```

```
    TAPE bitfield mask;
```

```
    SPOOL bitfield mask;
```

```
  end CJFFLGDEF_BITS;
```

```
end CJFFLGDEF;
```

```
end_module $CJFFLGDEF;
```

```
/* this is a tape based journal
```

```
/* the tape is being spooled at present
```

```
module $JNLBTXDEF;
```

```
/*++
/*
/* JNLBTX -      Journal block transfer
/*              This structure is used to define the offsets in the
/*              buffer allocated by CNX for our use with a block transfer.
/*
/*--
```

```
aggregate JNLBTXDEF structure fill prefix JNLBTX$:
  JNLBXSTS longword unsigned; /* Address if BXIP for this request
  RMBLK longword unsigned;   /* Address of Remaster block
  REQCSB longword unsigned;  /* Address of requestor's CSB
  SPARE1 longword unsigned;  /* Spare longword
  SPARE2 longword unsigned;  /* Spare longword
  SPARE3 longword unsigned;  /* Spare longword
  constant LENGTH equals . tag K ; /* Structure size
  constant LENGTH equals . tag C ; /* Structure size
```

```
end JNLBTXDEF;
```

```
end_module $JNLBTXDEF;
```



```
module $JNLDMTDEF;
```

```
/*++
```

```
/*
```

```
/* JNLDMT - codes for the parameters passed with dismount journal  
/* medium. These codes are used to identify the parameters  
/* to the Journal ACP, when passed in the complex buffer.  
/*
```

```
/*--
```

```
constant DNAM equals 1 prefix JNLDMT tag $C; /* device name parameter code  
constant DGRPN equals 2 prefix JNLDMT tag $C; /* group name parameter code  
constant FLAGS equals 3 prefix JNLDMT tag $C; /* flags value parameter code
```

```
end_module $JNLDMTDEF;
```

```
module $IOSTSDEF;
```

```
/*++
```

```
/*
```

```
/* IO status masks. These masks are in the third byte of IRP$L_IOSTS1  
/* and are used during a write operation to indicate  
/* the properties of the part (chunk) of the entry being written at the time,  
/* and the status of the IO request at certain times.  
/* The driver is the only one to use this I/O status field.
```

```
/*
```

```
/*--
```

```
aggregate IOSTSDEF union fill prefix IOSTS$;
```

```
    IOSTSDEF BITS structure fill;
```

```
        FSTCH bitfield mask;
```

```
        MULCH bitfield mask;
```

```
        SEQNOVF bitfield mask;
```

```
        WAITFIO bitfield mask;
```

```
        REMOTE bitfield mask;
```

```
        RESUB bitfield mask;
```

```
    end IOSTSDEF_BITS;
```

```
end IOSTSDEF;
```

```
end_module $IOSTSDEF;
```

```
/* First entry
```

```
/* Multiple entries
```

```
/* sequence number overflow
```

```
/* this IRP is waiting for buffer
```

```
/* write to complete.
```

```
/* This is an internal IRP and the
```

```
/* operation was started from a remote node
```

```
/* this request has not
```

```
/* been resubmitted yet if set
```

```
module $JNLMSGDEF;
```

```
/*++
```

```
/*
```

```
/* JNLMSG - JNLACP - Driver Cluster Message Definitions
```

```
/*
```

```
/*--
```

```
aggregate JNLMSGDEF structure fill prefix JNLMSG$;
```

```
FLINK longword unsigned; /* forward link
BLINK longword unsigned; /* backward link
SIZE word unsigned; /* size of structure
TYPE byte unsigned; /* structure type
SUBTYPE byte unsigned; /* structure sub-type
MSG_TYPE byte unsigned; /* message type
FILE_1 byte dimension 3 fill prefix JNLMSGDEF tag $$;
CSID longword unsigned; /* originator's CSID
```

```
constant 'HDRLEN' equals . prefix JNLMSG$ tag K; /* header length
constant 'HDRLEN' equals . prefix JNLMSG$ tag C; /* header length
```

```
constant (
    WRTBUFINF /* Write buffer information
    ,ALLDEV /* Add allocated device to ADL
    ,DEALLDEV /* Delete allocated device from ADL
    ,MNTDEV /* Add mounted device to ADL
    ,DMNTDEV /* Delete mounted device from ADL
    ,CRESLVDS /* Create slave data structures
    ) equals 1 increment 1 tag C;
```

```
end JNLMSGDEF;
```

```
/*
```

```
/* MESSAGE DEPENDENT EXTENSIONS
```

```
/*
```

```
/* MESSAGE 1 - Write buffer information
```

```
/*
```

```
aggregate JNLMSGDEF1 structure fill prefix JNLMSG$;
```

```
FILL_1 byte dimension JNLMSG$C_HDRLEN fill prefix JNLMSGDEF1 tag $$;
JNL_SEQN longword unsigned; /* Highest jnl seq # written to disk
LSEQNO longword unsigned; /* Lowest local seq # outstanding
SEQN_TCNT word unsigned; /* total # writes in CWQ for which jnl seq # have been
/* assigned (1 seq # per follows)
SEQN_CCNT word unsigned; /* current count of writes in CWQ for which jnl seq # have been
/* assigned (1 seq # per follows)
FILL_2 word unsigned fill prefix JNLMSGDEF1 tag $$; /* spare
```

```
constant MSG1_LEN equals . prefix JNLMSG$ tag K; /* Size of fixed part MSG1
constant MSG1_LEN equals . prefix JNLMSG$ tag C; /* Size of fixed part MSG1
```

```
end JNLMSGDEF1;
```

```
aggregate JNLMSGDEF1_SEQN structure fill prefix JNLMSG$;
```

```
/*
```

```
/* there is one of these JNLMSGDEF1_SEQN pieces per entry in the CWQ for
/* which a journal seq # has been assigned, in the message
```



```

/*
SEQ NUM longword unsigned;      /* Entry journal sequence number
FLAGS OVERLAY union fill;
  FCAGS longword unsigned;      /* flags longword
  FLAGS BITS structure fill;
    NEWVER bitfield mask;      /* Last write on a new version request
    PARTIAL bitfield mask;     /* Only part of the entry saved
end FLAGS BITS;
end FLAGS_OVERLAY;

constant "SEQENTLEN" equals . prefix JNLMSG$ tag K; /* length of sequence
constant "SEQENTLEN" equals . prefix JNLMSG$ tag C; /* number information

end JNLMSGDEF1_SEQN;

/*
/* MESSAGE DEPENDENT EXTENSIONS
/*
/* MESSAGE 2, 3, 4, 5, - Update the ADL
/*

aggregate JNLMSGDEF2 structure fill prefix JNLMSG$:
  FILL 1 byte dimension JNLMSG$C_HDRLEN fill prefix JNLMSGDEF2 tag $$;
  STATUS word unsigned;        /* status of device
  ITMLSTLEN word unsigned;      /* Item list length (mount only)
  ITMLSTOFF word unsigned;      /* Offset to item list (mount only)
  DEVNUM word unsigned;         /* # of dev names which follow
  NAMELEN byte unsigned;        /* device name length
  DEVNAM byte unsigned dimension 15; /* device name (ASCII)

  constant MSG2_LEN equals . prefix JNLMSG$ tag K; /* Size of fixed part MSG2
  constant MSG2_LEN equals . prefix JNLMSG$ tag C; /* Size of fixed part MSG2

end JNLMSGDEF2;

/*
/* MESSAGE DEPENDENT EXTENSIONS
/*
/* MESSAGE 6 - Create slave data structures
/* Each one byte item code in the message is followed by a longword which
/* is either the value or the offset to the information indicated.
/*

aggregate JNLMSGDEF6 structure fill prefix JNLMSG$:

  constant (
    BLDUCB          /* Build UCB - item value = journal type
    ,UCBDATA        /* offset to slave UCB data
    ,JNLNAM          /* offset to ASCII journal name
    ,BLDJNLRM        /* Build a remaster block - no item
    ,RMFLGS          /* JNLRM flags
    ,ACPNAM          /* offset to ASCII ACP name
    ,TAPGRP          /* offset to ASCII tape group name
    ,DSKINF          /* offset to ASCII disk name
    ,BLDADL          /* Build an ADL - no item

```

```
      .BLDRUL          /* Build an RUL - no item
      .MAXDSICOD       /* Maximum value
    } equals 1 increment 1 tag C;

ITEMCODE byte unsigned; /* Item code
ITEM longword unsigned; /* item information (value or offset)

constant IENTLEN equals . prefix JNLMSG$ tag C; /* Size of item entry
end JNLMSGDEF6;

aggregate JNLMSGDEF6_UCBDATA structure fill prefix JNLMSG$;

  OWNUIC longword unsigned; /* Owner UIC
  MCSID longword unsigned; /* Master CSID
  DEVCHAR longword unsigned; /* Device characteristics
  DEVCHAR2 longword unsigned; /* Device characteristics 2
  JNL_SEQNO longword unsigned; /* Journal sequence number
  JNL_QUOT longword unsigned; /* Quota for RU journals
  JNL_MASK longword unsigned; /* Mask for AT journals
  VPROT word unsigned; /* protection
  JNL_PROT word unsigned; /* protection
  JNL_ID word unsigned; /* Journal ID
  JNL_MXENT word unsigned; /* Maximum entry size
  JNL_MUNIT word unsigned; /* Master unit number
  DEVSTS word unsigned; /* Device status
  STS word unsigned; /* bits that need duplication on slave
  AMOD byte unsigned; /* Access mode

  constant UCBDATALEN equals . prefix JNLMSG$ tag C; /* Size of entry
end JNLMSGDEF6_UCBDATA;

end_module $JNLMSGDEF;
```

```
module $JNLMSGDATADEF;
```

```
/*++  
/*  
/* JNLMSGDATA -  
/*  
/*--
```

```
aggregate JNLMSGDATA structure fill prefix JNLMSGDATAS;
```

```
FLINK longword unsigned; /* Forward link  
BLINK longword unsigned; /* Backward link  
SIZE word unsigned; /* structure size  
TYPE byte unsigned; /* structure type code  
SUBTYPE byte unsigned; /* structure sub type field  
VAL1 longword unsigned; /* misc longword of data  
VAL2 longword unsigned; /* misc longword of data  
VAL3 longword unsigned; /* misc longword of data  
VAL4 longword unsigned; /* misc longword of data  
VAL5 longword unsigned; /* misc longword of data  
constant 'LENGTH' equals . prefix JNLMSGDATAS tag C;  
constant 'LENGTH' equals . prefix JNLMSGDATAS tag K;
```

```
end JNLMSGDATA;
```

```
end_module $JNLMSGDATADEF;
```



```
module $WBLDEF;
```

```
/*++
```

```
/*
```

```
/* WBL - Wait Block List
```

```
/*
```

```
/* When a thread is being rescheduled all its state is saved in a WBL
```

```
/*
```

```
/*--
```

```
aggregate WBLDEF structure fill prefix WBL$;
```

```
WBLQFL longword unsigned;
```

```
WBLQBL longword unsigned;
```

```
SIZE word unsigned;
```

```
FILL 1 word fill prefix WBLDEF tag $$;
```

```
STATUS longword unsigned;
```

```
ASTBLK longword unsigned;
```

```
IRP longword unsigned;
```

```
USTSIZE word unsigned;
```

```
FILL 2 word fill prefix WBLDEF tag $$;
```

```
UST longword unsigned;
```

```
USTADDR longword unsigned;
```

```
KSTSIZE word unsigned;
```

```
FILL 3 word fill prefix WBLDEF tag $$;
```

```
KST longword unsigned;
```

```
KSTADDR longword unsigned;
```

```
OWNSIZE word unsigned;
```

```
FILL 4 word fill prefix WBLDEF tag $$;
```

```
OWN longword unsigned;
```

```
FILL 5 longword fill prefix WBLDEF tag $$;
```

```
GBLSIZE word unsigned;
```

```
FILL 6 word fill prefix WBLDEF tag $$;
```

```
GBL longword unsigned;
```

```
FILL 7 longword fill prefix WBLDEF tag $$;
```

```
constant 'LENGTH' equals . prefix WBL$ tag K;
```

```
constant 'LENGTH' equals . prefix WBL$ tag C;
```

```
/* forward q link
```

```
/* backward q link
```

```
/* size of structure
```

```
/* spare
```

```
/* status
```

```
/* address AST Block for rescheduling
```

```
/* IRP address
```

```
/* user stack save block size
```

```
/* descriptor type field
```

```
/* address user stack save block
```

```
/* original start address user stack
```

```
/* kernel stack save block size
```

```
/* descriptor type field
```

```
/* address kernel stack save block
```

```
/* original start address kernel stack
```

```
/* own save block size
```

```
/* descriptor type field
```

```
/* address own space save block
```

```
/* spare
```

```
/* global save block size
```

```
/* descriptor type field
```

```
/* address global space save block
```

```
/* spare
```

```
/* length structure
```

```
/* length structure
```

```
end WBLDEF;
```

```
end_module $WBLDEF;
```

```
module $OPCHDRDEF;
```

```
/*++
/*
/* OPCHDR - OPCOM message header
/*
/* This structure defines the fields in the common OPCOM message
/* header. This data structure is defined in [SYS.SRC]SYSSNDMSG.MAR
/* in the commentary at the top. If this data structure ever changes in that
/* source module, then we need to change it here also.
/*
/*--
```

```
aggregate OPCHDRDEF structure fill prefix OPCHDR$;
```

```
TYPE word unsigned; /* message type
RMBX word unsigned; /* reply mailbox channel number
PRIV quadword unsigned; /* sender's privilege mask
UIC longword unsigned; /* sender's UIC
USRNAM byte unsigned dimension 12; /* sender's USERNAME, 12 bytes blank filled
ACCNT byte unsigned dimension 8; /* sender's ACCOUNT, 8 bytes blank filled
BPRIO byte unsigned; /* sender's base priority
FILL 1 byte fill prefix OPCHDRDEF tag $$; /* unused
constant 'LENGTH' equals . prefix OPCHDR$ tag K; /* length structure
constant 'LENGTH' equals . prefix OPCHDR$ tag C; /* length structure
end OPCHDRDEF;
```

```
end_module $OPCHDRDEF;
```

```
{ JNLSYSDEF : The following modules need to go into SYSDEF
```

```
{
{*****
{
{* Copyright (c) 1980
{* by DIGITAL Equipment Corporation, Maynard, Mass.
{*
{* This software is furnished under a license and may be used and copied
{* only in accordance with the terms of such license and with the
{* inclusion of the above copyright notice. This software or any other
{* copies thereof may not be provided or otherwise made available to any
{* other person. No title to and ownership of the software is hereby
{* transferred.
{*
{* The information in this software is subject to change without notice
{* and should not be construed as a commitment by DIGITAL Equipment
{* Corporation.
{*
{* DIGITAL assumes no responsibility for the use or reliability of its
{* software on equipment which is not supplied by DIGITAL.
{*****
{
```

```
{++
{ Facility: JOURNALING : DEFINITION OF INTERNAL SYMBOLICS
```

Abstract:

This module contains the symbolic definitions for non-user accessible data structures.

Author: Joost Verhofstad

18-MAY-1983

Modified by:

--


```
module $ABEDEF;
/****
/*
/* ABE - AI-BI List element
/*
/* For each AI or BI journal written to from inside an RU, the journal
/* name is in the AI-list or BI-list (for AI and BI journals resp)
/* This structure is the slot in the list, as used for one journal
/*
/*--

aggregate ABEDEF structure fill prefix ABES;
    JNLNAME character;                /* length name
    NAME byte unsigned dimension 18;   /* journal name
    STATUS_OVERLAY union fill;
        STATUS word unsigned;         /* status
        STATUS BITS structure fill;
            PURGED bitfield mask;      /* slot not used
        end STATUS BITS;
    end STATUS_OVERLAY;
    FILL 1 byte fill prefix ABEDEF tag $$;
    constant 'LENGTH' equals . prefix ABES tag K; /* spare
    constant 'LENGTH' equals . prefix ABES tag C; /* length structure
end ABEDEF;

end_module $ABEDEF;
```

```
module $ABLDEF;
```

```
/*++
```

```
/*
```

```
/* ABL - AI-BI List
```

```
/*
```

```
/* For each AI or BI journal written to from inside an RU, the journal
```

```
/* name is in the AI-List or BI-list (for AI and BI journals resp)
```

```
/*
```

```
/*--
```

```
aggregate ABLDEF structure fill prefix ABL$;
```

```
    NEXT longword unsigned;
```

```
    SLOTS word unsigned;
```

```
    JNLS word unsigned;
```

```
    SIZE word unsigned;
```

```
    TYPE OVERLAY union fill;
```

```
        STRUCT byte unsigned;
```

```
        TYPE byte unsigned;
```

```
    end TYPE OVERLAY;
```

```
    SUBTYPE byte unsigned;
```

```
    constant FIXED_LEN equals . prefix ABL$ tag K;
```

```
    constant FIXED_LEN equals . prefix ABL$ tag C;
```

```
end ABLDEF;
```

```
end_module $ABLDEF;
```

```
/* next ABL
```

```
/* number of slots in list
```

```
/* number of journals in list
```

```
/* size structure
```

```
/* structure type
```

```
/* data type field
```

```
/* CJF subtype field
```

```
/* length structure
```

```
/* length structure
```

```

module $ADBDEF;
/*++
/*
/* ADB - Allocated Device Block
/*
/* For each disk or tape device allocated by a Journal ACP, the
/* ADL off the UCB for the ACP Control Journal contains a ADB
/* (Allocated Device Block). The ADB contains the device name
/* and some control information
/*
/*--

aggregate ADBDEF structure fill prefix ADB$:
    LINK longword unsigned;          /* link to next ADB in same volume set
    STATUS_OVERLAY union fill;
        STATUS word unsigned;        /* status of device and this ADB
        STATUS_BITS structure fill;
            MNTALLOC bitfield mask;  /* allocated during MOUNT
            MOUNTED bitfield mask;   /* device is mounted
            PURGED bitfield mask;    /* this ADB is available
        end STATUS_BITS;
    end STATUS_OVERLAY;
    FILL_1 word fill prefix ADBDEF tag $$;
    FILL_2 longword fill prefix ADBDEF tag $$;
    NAMELEN byte unsigned;
    DEVNAM byte unsigned dimension 15;
    constant 'LENGTH' equals . prefix ADB$ tag K;
    constant 'LENGTH' equals . prefix ADB$ tag C;
end ADBDEF;

end_module $ADBDEF;

```



```
module $ADLDEF;
```

```
/*++
```

```
/*
```

```
/* ADL - Allocated Device List
```

```
/*
```

```
/* For each disk or tape device allocated by a Journal ACP, the
```

```
/* ADL off the UCB for the ACP Control Journal contains a ADB
```

```
/* (Allocated Device Block).
```

```
/*
```

```
/*--
```

```
aggregate ADLDEF structure fill prefix ADL$;
```

```
LINK longword unsigned;
```

```
UCB longword unsigned;
```

```
SIZE word unsigned;
```

```
TYPE byte unsigned;
```

```
SUBTYPE byte unsigned;
```

```
EXTEND longword unsigned;
```

```
DEVCNT word unsigned;
```

```
ADBCNT word unsigned;
```

```
FSTADB word unsigned;
```

```
FILL_2 word fill prefix ADLDEF tag $$;
```

```
constant FIXED_LEN equals . prefix ADL$ tag K;
```

```
constant FIXED_LEN equals . prefix ADL$ tag C;
```

```
constant START_ADB equals . prefix ADL$ tag K;
```

```
constant START_ADB equals . prefix ADL$ tag C;
```

```
end ADLDEF;
```

```
end_module $ADLDEF;
```

```
/* Link to next ADL for this ACP (only
```

```
/* for first ADL, not for extensions)
```

```
/* backpointer to UCB
```

```
/* size of list (ADL+ADB in this ADL)
```

```
/* data structure type
```

```
/* CJF subtype field
```

```
/* next ADL extension
```

```
/* device count: ! of devices allocated
```

```
/* in this ADL
```

```
/* number of ADBs in this ADL
```

```
/* offset first ADB, from this location
```

```
/* spare
```

```
/* length fixed portion
```

```
/* length fixed portion
```

```
/* Start of list.
```

```
/* Start of list.
```

```
module $BCBDEF;
```

```
/*++
```

```
/*
/* BCB - Buffer Control Block
/*
/* For each mounted journal there are two buffers pointed to by the
/* BCB which is pointed to by the journal UCB. The BCB always describes
/* the characteristics and status of these buffers
/*
/*--
```

```
aggregate BCBDEF structure fill prefix BCBS;
```

```
  ADDR1 longword unsigned; /* address of buffer 1
  ADDR2 longword unsigned; /* address of buffer 2
  SIZE word unsigned; /* structure size
  TYPE byte unsigned; /* structure type code
  SUBTYPE byte unsigned; /* subtype field for CJF
  STS_OVERLAY union fill;
    STS byte unsigned; /* status code
    STS_BITS structure fill;
      CUR bitfield mask; /* current buffer indicator
    end STS_BITS;
  end STS_OVERLAY;
  FILL_1 word fill prefix BCBDEF tag $$; /* SPARE
  FILL_2 byte fill prefix BCBDEF tag $$; /* SPARE
  UCB longword unsigned; /* UCB address of journal
  BSIZ1 word unsigned; /* size of buffer 1 in bytes
  BSIZ2 word unsigned; /* size of buffer 2 in bytes
  STS1_OVERLAY union fill;
    STS1 word unsigned; /* status of buffer 1
    STS1_BITS structure fill;
      TOPR bitfield mask; /* I/O in progress bit
      WRPR bitfield mask; /* write in progress bit
      WRPEN bitfield mask; /* write pending bit
      REPR bitfield mask; /* read in progress bit
      REAPEN bitfield mask; /* read pending bit
      EXTPR bitfield mask; /* extend in progress
      EXTPEN bitfield mask; /* extend pending
      RECLE bitfield mask; /* buffer read and cleared bit
      SETPEN bitfield mask; /* "set-buffer-to-next-one" pending
      NWVPR bitfield mask; /* create new version in progress
    end STS1_BITS;
  end STS1_OVERLAY;
  STS2 word unsigned; /* status of buffer 2
  WRCNT1 word unsigned; /* write count for first buffer
  WRCNT2 word unsigned; /* write count for second buffer
  RDCNT1 word unsigned; /* read count for first buffer
  RDCNT2 word unsigned; /* read count for second buffer
  OFFS1 word unsigned; /* offset first free byte in buffer 1
  OFFS2 word unsigned; /* offset first free byte in buffer 2
  VBN1 longword unsigned; /* first VBN buffer 1
  VBN2 longword unsigned; /* first VBN buffer 2
  PRVVBN longword unsigned; /* VBN bucket in which previous chunk is
  PRVEVBN longword unsigned; /* VBN bucket in which previous entry is
  PRVOFF word unsigned; /* offset of previous chunk written
```

```
PRVEOFF word unsigned; /* offset of previous entry written
LOWSN longword unsigned; /* lowest seq.no in current buffer
HISN longword unsigned; /* highest seq.no of any entry written
                          /* into the buffers
CRCTBL longword unsigned; /* address of CRC table
HISN_CMPL longword unsigned; /* High sequence number completely in a buffer
HISN_WRT longword unsigned; /* High sequence number written
                          /* to secondary storage
constant 'LENGTH' equals . prefix BCBS tag K; /* length of structure
constant 'LENGTH' equals . prefix BCBS tag C; /* length of structure
end BCBDEF;
end_module $BCBDEF;
```

```
module $JNLACBMDEF;
```

```
/*++
```

```
/*
```

```
/* JNLACBM - Journal access bit map
```

```
/*
```

```
/* This bit map will contain a single bit for each node in  
/* the cluster. When ever a slave node assigns his first  
/* journal to the journal or deassigns his last journal channel  
/* the node bit will be adjusted. This bit map will be indexed  
/* via the node index portion of the node's CSID
```

```
/*--
```

```
aggregate JNLACBMDEF structure fill prefix JNLACBMS;
```

```
FLINK longword unsigned;
```

```
/* Forward link
```

```
BLINK longword unsigned;
```

```
/* Backward link
```

```
SIZE word unsigned;
```

```
/* structure size
```

```
TYPE byte unsigned;
```

```
/* structure type code
```

```
SUBTYPE byte unsigned;
```

```
/* structure sub type field
```

```
MAPSIZE word unsigned;
```

```
/* Bit map size
```

```
BITMAP character length 0 tag X ;
```

```
/* Bit map start
```

```
constant LENGTH equals .;
```

```
/* Size of JNLACBM header
```

```
end JNLACBMDEF ;
```

```
end_module $JNLACBMDEF ;
```



```
module $JNLBUFDEF;
```

```
/*++
```

```
/*
```

```
/* JNLBUF - Buffer of which there are two for each journal
```

```
/*
```

```
/* The BCB pointed to by the journal UCB points to the two buffers
```

```
/*
```

```
/*--
```

```
aggregate JNLBUFDEF structure fill prefix JNLBUF$;
```

```
LEN word unsigned;
```

```
LEN2 word unsigned;
```

```
TYPE_OVERLAY union fill;
```

```
TYPE byte unsigned;
```

```
TYPE BITS structure fill;
```

```
USER bitfield mask;
```

```
CONTR bitfield mask;
```

```
end TYPE BITS;
```

```
end TYPE_OVERLAY;
```

```
BUFHDR byte unsigned;
```

```
FILL 1 word fill prefix JNLBUFDEF tag $$;
```

```
BUFSTZ word unsigned;
```

```
DTYPE_OVERLAY union fill;
```

```
STRUCT byte unsigned;
```

```
DTYPE byte unsigned;
```

```
end DTYPE_OVERLAY;
```

```
STYPE_OVERLAY union fill;
```

```
ENTTYP byte unsigned;
```

```
SUBTYPE byte unsigned;
```

```
end STYPE_OVERLAY;
```

```
VBN longword unsigned;
```

```
LSTEND word unsigned;
```

```
FILL 2 word fill prefix JNLBUFDEF tag $$;
```

```
JNLID longword unsigned;
```

```
LOWSN longword unsigned;
```

```
HISN longword unsigned;
```

```
CDPTR word unsigned;
```

```
STS_OVERLAY union fill;
```

```
STS word unsigned;
```

```
STS_BITS structure fill;
```

```
UPDATE bitfield mask;
```

```
end STS_BITS;
```

```
end STS_OVERLAY;
```

```
CHKSUM longword unsigned;
```

```
constant HDRLEN equals . prefix JNLBUF$ tag K;
```

```
constant HDRLEN equals . prefix JNLBUF$ tag C;
```

```
constant STDAT equals . prefix JNLBUF$ tag K;
```

```
constant STDAT equals . prefix JNLBUF$ tag C;
```

```
/* total length of buffer header minus
```

```
/* length of this word (RMS seq. record)
```

```
/* second word of length (only for tape)
```

```
/* record type to indicate control entry
```

```
/* user entry
```

```
/* control entry
```

```
/* buffer header length
```

```
/* SPARE (to match other records)
```

```
/* buffer size : this MUST be 1st word
```

```
/* in 3rd longword
```

```
/* data structure type value : this MUST
```

```
/* be 3rd byte in 3rd longword
```

```
/* data type field
```

```
/* entry type
```

```
/* data subtype field
```

```
/* journal block number (of 1st. bl in bucket)
```

```
/* last entry/chunk in bucket - offset
```

```
/* spare
```

```
/* journal ID
```

```
/* lowest sequence number of all entries
```

```
/* in this bucket
```

```
/* highest sequence number of all entries
```

```
/* in this bucket
```

```
/* current data pointer (! of data bytes
```

```
/* written for BI, AI, AT and next byte
```

```
/* to write for RU jnl)
```

```
/* buffer status
```

```
/* this buffer has been updated
```

```
/* CRC of bucket
```

```
/* length header
```

```
/* length header
```

```
/* first longword of data
```

```
/* first longword of data
```

end JNLBUFDEF;

end_module \$JNLBUFDEF;

```
module $JNLBXSTSDEF;
```

```
/*++
```

```
/*
/* JNLBXSTS - Journal block transfer in process queue entry
/* This structure is used to keep track of all pertinent
/* information concerning an IRP that has been initiated
/* on the local node via a block transfer request from
/* some other node. If the connection between the two
/* nodes breaks before the local node has sent the response
/* then the message may be retransmitted and we must be
/* able to deal with that. Hopefully this structure will
/* contain all the information we will need.
/*
/*--
```

```
/*--
```

```
aggregate JNLBXSTSDEF structure fill prefix JNLBXST$:
```

```
FLINK    longword unsigned;    /* Forward link
BLINK    longword unsigned;    /* Backward link
SIZE     word unsigned;        /* size data structure
TYPE     byte unsigned;        /* type of structure
SUBTYPE  byte unsigned;        /* subtype of structure
STS_OVERLAY union fill;
  STS     longword unsigned;    /* block Xfer status
  STS_BITS structure fill;
    READCMPL bitfield mask;    /* The block read is complete
    READINP  bitfield mask;    /* The block read is in progress
    WRITECMPL bitfield mask;    /* The block write is complete
    WRITEINP  bitfield mask;    /* The block write is in progress
    RESPSNT  bitfield mask;    /* The response has been sent
    FNCTCMPL bitfield mask;    /* The function is complete (no response sent)
    CNXBRK   bitfield mask;    /* The connection has broken
  end STS_BITS;
end STS_OVERLAY;
REQ_CSID_OVERLAY union fill;
  REQ_CSID longword unsigned;    /* CSID of node which originated
                                   /* the message (requestor)
  REQ_CSID_SUBF structure fill;
    REQ_CSID_SEQ word unsigned;  /* CSID sequence number
    REQ_CSID_IDX word unsigned;  /* CSID node index
  end REQ_CSID_SUBF;
end REQ_CSID_OVERLAY;
BTXSEQNO longword unsigned;    /* Block transfer sequence #
CURR_IRP longword unsigned;    /* Address of the current IRP
RTX_IRP longword unsigned;    /* Address of IRP from last retransmit
SPARE1 longword unsigned;
SPARE2 longword unsigned;
SPARE3 longword unsigned;
constant LENGTH equals . tag K; /* Structure size
constant LENGTH equals . tag C; /* Structure size
```

```
end JNLBXSTSDEF;
```

```
end_module $JNLBXSTSDEF;
```


WRATR byte unsigned;	/* Write attributes
EPID longword unsigned;	/* Process EPID
ARB PRIV quadword unsigned;	/* Priv mask from ARB
MSGBUF character length 0;	/* Base of journal entry in a message
constant FIXED_LEN equals . tag C ;	/* Fixed size
constant FIXED_LEN equals . tag K ;	/* Fixed size

end JNLCWQDEF;

end_module \$JNLCWQDEF;

```
module $JNLDBDEF;
```

```
/*++
/*
/* JNLDB - off of each CDT is hung a data block that serves as
/* a queue listhead for remote IRP's waiting on a response
/* for a connection, a queue listhead for the slave UCB's
/* that access the master node via that CDT, and a pointer
/* to a buffer that contains entries written to the master
/* (via the CDT) but whose QIOs have not yet been ACK'd
/* by the master. This structure is used for master
/* failover recovery.
/*--
```

```
aggregate JNLDBDEF structure fill prefix JNLDB$;
```

```
IRPQFL longword unsigned;
```

```
IRPQBL longword unsigned;
```

```
SIZE word unsigned;
```

```
TYPE byte unsigned;
```

```
SUBTYPE byte unsigned;
```

```
UCBQFL longword unsigned;
```

```
UCBQBL longword unsigned;
```

```
BUFFER longword unsigned;
```

```
FILL 1 longword fill prefix JNLDBDEF tag $$;
```

```
constant 'LENGTH' equals . prefix JNLDB$ tag K;
```

```
constant 'LENGTH' equals . prefix JNLDB$ tag C;
```

```
end JNLDBDEF;
```

```
end_module $JNLDBDEF;
```

```
/* IRP queue forward link
/* IRP queue backward link
/* size data structure
/* type of structure
/* subtype of structure
/* UCB queue forward link
/* UCB queue backward link
/* Pointer to write buffer
/* Spare
```

```
module $JNLLOGDEF;
```

```
/*++
```

```
/*
```

```
/* JNLLOG - Journal error log function bits
```

```
/*
```

```
/* This structure defines the bits indicating to SYE the error
```

```
/* being logged
```

```
/*
```

```
/*--
```

```
aggregate JNLLOGDEF union fill prefix JNLLOG$;
```

```
    JNLLOGDEF_BITS structure fill;
```

```
        RUEXT bitfield mask;
```

```
        /* RU journal extended
```

```
        RUNEXT bitfield mask;
```

```
        /* RU journal could not be extended
```

```
        SLVCRFAIL bitfield mask;
```

```
        /* Failure on slave node while
```

```
        /* attempting a create
```

```
    end JNLLOGDEF_BITS;
```

```
end JNLLOGDEF;
```

```
end_module $JNLLOGDEF;
```

```
module $JNLRCDEF;
```

```
/*++
```

```
/*
```

```
/* JNLRC - Journaling Read Context
```

```
/*
```

```
/* The JNLRC holds the information necessary for read failover.
```

```
/*
```

```
/*--
```

```
aggregate JNLRCDEF structure fill prefix JNLRC$;
```

```
  FILL_1 longword fill prefix JNLRCDEF tag $$;
```

```
  FILL_2 longword fill prefix JNLRCDEF tag $$;
```

```
  SIZE word unsigned;
```

```
  TYPE byte unsigned;
```

```
  SUBTYPE byte unsigned;
```

```
  SEQNO longword unsigned;
```

```
  RUID UNION union fill;
```

```
    RUID quadword unsigned octaword;
```

```
    RUID OVERLAY structure fill;
```

```
      DATTIM quadword unsigned;
```

```
      CSID UNION union fill;
```

```
        CSID longword unsigned;
```

```
        CSID OVERLAY structure fill;
```

```
          CSID_SEQ word unsigned;
```

```
          CSID_IDX word unsigned;
```

```
        end CSID OVERLAY;
```

```
      end CSID UNION;
```

```
      RUID_LW4 longword unsigned;
```

```
    end RUID OVERLAY;
```

```
  end RUID UNION;
```

```
  FLAGS OVERLAY union fill;
```

```
    FLAGS byte unsigned;
```

```
    FLAGS BITS structure fill;
```

```
      READDR bitfield mask;
```

```
    end FLAGS BITS;
```

```
  end FLAGS OVERLAY;
```

```
  FILL_3 byte dimension 3 fill prefix JNLRCDEF tag $$; /* spare
```

```
  FLTRS longword unsigned; /* Offset to filters
```

```
  constant 'LENGTH' equals . prefix JNLRC$ tag K; /* length fixed part
```

```
  constant 'LENGTH' equals . prefix JNLRC$ tag C; /* length fixed part
```

```
/* unused - forward link
```

```
/* unused - back link
```

```
/* size of structure
```

```
/* data structure type
```

```
/* CJF subtype
```

```
/* seq # previous entry
```

```
/* Recovery unit ID (RU only)
```

```
/* date/time prev. entry (NONRU ONLY)
```

```
/* CSID portion of RUID,
```

```
/* CSID sequence number
```

```
/* CSID node index
```

```
/* Forth longword of RUID
```

```
/* Flags
```

```
/* Read direction
```

```
end JNLRCDEF;
```

```
end_module $JNLRCDEF;
```

```
module $JNLRMDEF;
```

```
/*++
```

```
/*
```

```
/* JNLRM - Journaling Remaster Block
```

```
/*
```

```
/* The JNLRM is used by the CSP to construct a JSB for remastering a journal.
```

```
/*
```

```
/*--
```

```
aggregate JNLRMDEF structure fill prefix JNLRMS;
```

```
  FILL_1 longword fill prefix JNLRMDEF tag $$;
```

```
  FILL_2 longword fill prefix JNLRMDEF tag $$;
```

```
  SIZE word unsigned;
```

```
  TYPE byte unsigned;
```

```
  SUBTYPE byte unsigned;
```

```
  FLAGS OVERLAY union fill;
```

```
    FLAGS word unsigned;
```

```
    FLAGS BITS structure fill;
```

```
      DSKJNL bitfield mask;
```

```
      TAPJNL bitfield mask;
```

```
      TMPFIL bitfield mask;
```

```
      DIFACP bitfield mask;
```

```
    end FLAGS BITS;
```

```
  end FLAGS OVERLAY;
```

```
  COPIES byte unsigned;
```

```
  FILL_3 byte fill prefix JNLRMDEF tag $$;
```

```
  CONBCK longword unsigned;
```

```
  ACPNAMOFF word unsigned;
```

```
  ACPNAMLEN word unsigned;
```

```
  constant "LENGTH" equals . prefix JNLRMS tag K;
```

```
  constant "LENGTH" equals . prefix JNLRMS tag C;
```

```
  constant "DSKJNLLST" equals . prefix JNLRMS tag K;
```

```
  constant "DSKJNLLST" equals . prefix JNLRMS tag C;
```

```
  TAPGRPOFF word unsigned;
```

```
  TAPGRPLEN word unsigned;
```

```
  constant "TAPJNLLEN" equals . prefix JNLRMS tag K;
```

```
  constant "TAPJNLLEN" equals . prefix JNLRMS tag C;
```

```
/* unused - forward link
```

```
/* unused - backward link
```

```
/* size of structure
```

```
/* data structure type
```

```
/* subtype for CJF data structure
```

```
/* flags word
```

```
/* Disk journal
```

```
/* Tape journal
```

```
/* Temp file
```

```
/* Different ACP
```

```
/* number of copies
```

```
/* fill
```

```
/* address of the 1st connect block
```

```
/* offset to ACP name
```

```
/* ACP name length
```

```
/* length fixed part
```

```
/* length fixed part
```

```
/* start info for disk jnls
```

```
/* - dev names, ver #'s
```

```
/* offset to tape group name
```

```
/* tape group name length
```

```
/* length for tape journal
```

```
/* length for tape journals
```

```
end JNLRMDEF;
```

```
aggregate JNLRM1DEF structure fill prefix JNLRMS;
```

```
  DEVNAMOFF word unsigned;
```

```
  DEVNAMLEN word unsigned;
```

```
  FILVEROFF word unsigned;
```

```
  FILVERLEN word unsigned;
```

```
/* offset to device name
```

```
/* device name length
```

```
/* offset to file version
```

```
/* file version length
```

```
  constant "DSKENTLEN" equals . prefix JNLRMS tag K;
```

```
  constant "DSKENTLEN" equals . prefix JNLRMS tag C;
```

```
/* length of disk journal
```

```
/* information
```


JNLDEFINT.SDL;1

16-SEP-1984 16:40:05.92^{5.12} Page 33

end JNLRM1DEF;

end_module \$JNLRMDEF;

```
module $JNLSFTDEF;
```

```
/*
```

```
/* JNLSFT -- Spool File Table
```

```
/*
```

```
/* The JNLSFT describes the physical storage medium for the journal spool
```

```
/* file. Spool files are used for tape groups only.
```

```
/* The JNLSFTs for a given tape group are linked together in a list.
```

```
/* The first JNLSFT is pointed to by each JMT for each tape in the group.
```

```
/*
```

```
/*-
```

```
aggregate JNLSFTDEF structure fill prefix JNLSFT$;
```

```
FORJNLLNK longword unsigned;
```

```
BACJNLLNK longword unsigned;
```

```
SIZE word unsigned;
```

```
TYPE byte unsigned;
```

```
SUBTYPE byte unsigned;
```

```
constant ACPQB equals . prefix JNLSFT$ tag K;
```

```
constant ACPQB equals . prefix JNLSFT$ tag C;
```

```
FORACPLNK longword unsigned;
```

```
BACACPLNK longword unsigned;
```

```
JMT longword unsigned;
```

```
SPL_COP byte unsigned;
```

```
FILC_2 byte dimension 3 fill prefix JNLSFTDEF tag $$; /* spare
```

```
MAX_JNLS word unsigned;
```

```
COPY_NUM word unsigned;
```

```
WRCNT word unsigned;
```

```
RDCNT word unsigned;
```

```
STATUS_OVERLAY union fill;
```

```
STATUS longword unsigned;
```

```
STATUS_BITS structure fill;
```

```
HEAD_SFT bitfield mask;
```

```
ACTIVE bitfield mask;
```

```
end STATUS_BITS;
```

```
end STATUS_OVERLAY;
```

```
BASEVBN longword unsigned;
```

```
SPL_UCB longword unsigned;
```

```
SPL_UCB longword unsigned;
```

```
SPL_MXVBN longword unsigned;
```

```
SPL_STVBN longword unsigned;
```

```
SPL_NUM word unsigned;
```

```
SPL_SEQ word unsigned;
```

```
SPL_RVN word unsigned;
```

```
FILC_3 word fill prefix JNLSFTDEF tag $$;
```

```
VOLLAB byte unsigned dimension 12;
```

```
SPL_VBN longword unsigned;
```

```
constant 'LENGTH' equals . prefix JNLSFT$ tag K;
```

```
constant 'LENGTH' equals . prefix JNLSFT$ tag C;
```

```
/* Forward link for JMT's for this journal
```

```
/* Backward link for JMT's for this journal
```

```
/* size of JNLSFT
```

```
/* structure type of JNLSFT
```

```
/* structure subtype of JNLSFT
```

```
/* label for ACP queue block
```

```
/* label for ACP queue block
```

```
/* Forward link to next JMT for this ACP
```

```
/* Backward link to next JMT for this ACP
```

```
/* First JMT in list of JMTs for group
```

```
/* for which this is a spool file
```

```
/* number of spool files in list
```

```
/* spare
```

```
/* max ! of journals for this spool file
```

```
/* number of spool file (zero relative)
```

```
/* write count
```

```
/* read count
```

```
/* journal media status
```

```
/* first JNLSFT (copy) for this group
```

```
/* spool file not empty: being used
```

```
/* Base VBN: to be subtracted from bucket
```

```
/* VBN to get VBN of block in file
```

```
/* pointer to journal spool file WCB
```

```
/* pointer to journal spool file UCB
```

```
/* max VBN in journal disk spool file
```

```
/* first VBN in journal disk spool file
```

```
/* journal spool file file ID number
```

```
/* journal spool file file ID sequence number
```

```
/* journal spool file file ID rel vol num
```

```
/* spare
```

```
/* volume label disk on which file is
```

```
/* next VBN for next bucket to write to
```

```
/* length
```

```
/* length
```

```
/* spool file. (spool file is used as
```

```
/* tape, but we must keep track of VBN)
```

JNLDEFINT.SDL;1

16-SEP-1984 16:40:05.94^{M 12} Page 35

end JNLSFTDEF;

end_module \$JNLSFTDEF;

```
module SJMTDEF;
```

```
/*  
/* JMT -- Journal Merge Table  
/*  
/* The JMT describes the physical storage medium for the journal copy.  
/* The JMT is pointed to by each VCB. When multiple journals are  
/* kept on the same storage medium (ie multiple journals on one  
/* tape), there exists one JMT for the tape, and many VCB's may  
/* point to it.  
/*  
/* All bits marked (*) are set in the head JMT (first in list) only  
/* in the current version.  
/*  
/*-
```

```
aggregate JMTDEF structure fill prefix JMT$;
```

```
FORJNLLNK longword unsigned; /* Forward link for JMT's for this journal  
BACJNLLNK longword unsigned; /* Backward link for JMT's for this journal  
SIZE word unsigned; /* size of JMT  
TYPE byte unsigned; /* structure type of JMT  
SUBTYPE byte unsigned; /* structure subtype of JMT  
constant ACPQB equals . prefix JMT$ tag K; /* label for ACP queue block  
constant ACPQB equals ; prefix JMT$ tag C; /* label for ACP queue block  
FORACPLNK longword unsigned; /* Forward link to next JMT for this ACP  
BACACPLNK longword unsigned; /* Backward link to next JMT for this ACP
```

```
ACP PRI byte unsigned; /* ACP's priority (priority for I/O)  
FILE_2 byte dimension 3 fill prefix JMTDEF tag $$; /* spare  
ACP_ARB longword unsigned; /* pointer to ACP access rights block  
AQB longword unsigned; /* address of AQB for owner ACP
```

```
MAX JNLS word unsigned; /* max ! of journals for this JMT  
FILE_3 word fill prefix JMTDEF tag $$; /* spare  
COPY_NUM word unsigned; /* copy number (zero relative)  
JNLIDCTR word unsigned; /* journal ID counter  
WRCNT word unsigned; /* write count  
RDCNT word unsigned; /* read count
```

```
SPOOLING_OVERLAY union fill;  
SPOOLING byte unsigned; /* spool byte: if any of these bits is  
/* set, spooling must be done.
```

```
SPOOLING_BITS structure fill;  
REPR bitfield mask; /* read in progress  
EOTPR bitfield mask; /* EOT processing going on (*)  
end SPOOLING_BITS;
```

```
end SPOOLING_OVERLAY;  
FILL_4 byte dimension 3 fill prefix JMTDEF tag $$; /* spare
```

```
end JMTDEF;
```

```
aggregate JMTDEF1 structure fill prefix JMT$;
```

```
FILL_10 byte dimension 44 fill prefix JMTDEF tag $$;  
STATUS_OVERLAY union fill;  
STATUS longword unsigned; /* journal media status
```

```

STATUS BITS structure fill;
  SPCBYTE bitfield mask length 8;
  WRPR bitfield mask;
  NOWRJNL bitfield mask;

  HEAD JMT bitfield mask;
  SPOOLED bitfield mask;
  SPOOLSYNC bitfield mask;

  STARTSP bitfield mask;
  STOPSP bitfield mask;
  CANCELIO bitfield mask;
  DMT bitfield mask;
  AVL bitfield mask;
  SYNCHCAN bitfield mask;
  REPEN bitfield mask;
  INFPEN bitfield mask;
  NOWRTP bitfield mask;
end STATUS BITS;
end STATUS_OVERLAY;
JMTSFT longword unsigned;

SPARE1 longword unsigned;
SPARE2 longword unsigned;
SPARE3 longword unsigned;
SPARE4 longword unsigned;
OWNUIC longword unsigned;
PROT word unsigned;
FILL_5 word fill prefix JMTDEF tag $$;

BASEVBN longword unsigned;
FIL_WCB longword unsigned;
FIL_UCB longword unsigned;
FIL_MXVBN longword unsigned;
FIL_STVBN longword unsigned;
FIL_LTVBN longword unsigned;
FIL_NUM word unsigned;
FIL_SEQ word unsigned;
FIL_RVN word unsigned;
FILL_6 word fill prefix JMTDEF tag $$;
VOLLAB character length 13;
FILL_7 byte fill prefix JMTDEF tag $$;
GRPHAM character length 13;
FILL_8 byte fill prefix JMTDEF tag $$;
GTB longword unsigned;

JFTE longword unsigned;

SFT longword unsigned;
SPL_VBN longword unsigned;

VCB_COUNT word unsigned;

/* spool byte
/* write in progress (currently unused)
/* cannot write to journal now (not
/*   even spool file)
/* first JMT (copy) for this journal
/* device is spooled (*)
/* all io to journal file (incl spool
/*   file) must wait: switching back or
/*   forth between tape and spool file
/*   (*)
/* start spooling (*)
/* stop spooling (*)
/* cancel IO to tape (*)
/* this copy is marked for dismount
/* this copy is available
/* synchronize with CANCELIO on tape (*)
/* read pending
/* inform ACP pending (*)
/* do not write to tape: ACP stops driver

/* the JMT or SFT on which an error
/* occurred (*)

/* owner UIC
/* protection mask
/* SPARE

/* base VBN first bucket (add to file VBN to get bucket VBN)
/* pointer to journal file WCB
/* pointer to journal file UCB
/* max VBN in journal disk file
/* first VBN in journal disk file
/* last VBN for this file
/* journal file file ID number
/* journal file file ID sequence number
/* journal file file ID rel vol num
/* spare
/* volume label disk/tape on which file is
/* spare
/* group name
/* spare
/* address of corresponding GTB in ACP
/*   virtual memory
/* address of corresponding JFTE in ACP
/*   virtual memory
/* first SFT (spool file table)
/* next VBN for next bucket to write to
/* spool file. (spool file is used as
/* tape, but we must keep track of VBN)

/* number of VCB's pointing to JMT

```



```
FILL 9 word fill prefix JMTDEF tag $$;
VCB_CNTRL longword unsigned;
WQFC longword unsigned;
WQBL longword unsigned;
VCL longword unsigned;
FILVER character length 6;
constant 'LENGTH' equals . prefix JMT$ tag K;
constant 'LENGTH' equals . prefix JMT$ tag C;
end JMTDEF1;

end_module $JMTDEF;
```

/* (not including VCB_CNTRL)
/* spare
/* address of control VCB (tape only)
/* wait 0 forward link
/* wait 0 backward link
/* list of addresses associated VCB's
/* file version number
/* length label
/* length label

```
module $NDLDEF;
```

```
/*++
```

```
/*
```

```
/* NDL - Name table Device List
```

```
/*
```

```
/* This structure has a fixed header size but the tail end is a variable
```

```
/* length depending on how many name table device names are in it.
```

```
/*
```

```
/*--
```

```
aggregate NDLDEF structure fill prefix NDL$;
```

```
NDLOFL longword unsigned;
```

```
/* forward q link
```

```
NDLOBL longword unsigned;
```

```
/* backward q link
```

```
SIZE word unsigned;
```

```
/* size of structure
```

```
TYPE byte unsigned;
```

```
/* structure type for NDL
```

```
SUBTYPE byte unsigned;
```

```
/* structure subtype
```

```
COUNT byte unsigned;
```

```
/* count
```

```
FILL_1 word fill prefix NDLDEF tag $$;
```

```
/* spare
```

```
FILL_2 byte fill prefix NDLDEF tag $$;
```

```
/* spare
```

```
constant FIXEDLEN equals . prefix NDL$ tag K;
```

```
/* fixed size length
```

```
constant FIXEDLEN equals . prefix NDL$ tag C;
```

```
/* fixed size length
```

```
end NDLDEF;
```

```
end_module $NDLDEF;
```



```
        RUSYNCEX bitfield mask;  
        RUSYNCWR bitfield mask;  
        NOFAC bitfield mask;  
        NOOBJ bitfield mask;  
    end STATUS BITS;  
end STATUS_OVERLAY;  
end RUEDEF;  
end_module $RUEDEF;
```

```
/* RUSYNC entry expected  
/* RUSYNC entry written  
/* Frozen due to missing facility  
/* Frozen due to missing object
```

```
module $RULDEF;
```

```
/*++
```

```
/*
```

```
/* RUL - Recovery Unit List
```

```
/*
```

```
/* This data structure forms the header of the list with the recovery
```

```
/* units that are currently active on the RU-journal for which this
```

```
/* list is used. The UCB of a RU journal points to the RUL for it.
```

```
/*
```

```
/*--
```

```
aggregate RULDEF structure fill prefix RUL$;
```

```
    NUM_RUES word unsigned;
```

```
    FILL_1 word fill prefix RULDEF tag $$;
```

```
    FILL_2 longword fill prefix RULDEF tag $$;
```

```
    SIZE word unsigned;
```

```
    TYPE byte unsigned;
```

```
    SUBTYPE byte unsigned;
```

```
    constant FIXED_LEN equals . prefix RUL$ tag K;
```

```
    constant FIXED_LEN equals . prefix RUL$ tag C;
```

```
/* number of RUEs in the list
```

```
/* spare
```

```
/* spare
```

```
/* size of total list (RUL+all RUEs)
```

```
/* data structure type
```

```
/* data structure subtype
```

```
/* length of RUL fixed portion
```

```
/* length of RUL fixed portion
```

```
end RULDEF;
```

```
end_module $RULDEF;
```



```
module $VCLDEF;
```

```
/*+
```

```
/* VCL - VCB List
```

```
/*
```

```
/* The VCL contains the VCB addresses of VCBs of journals that have been  
/* created for a given tape group. The JMT of the head-JMT for that group  
/* points to this VCL.
```

```
/*
```

```
/*-
```

```
aggregate VCLDEF structure fill prefix VCLS;
```

```
    JMT longword unsigned;
```

```
    NUM VLES word unsigned;
```

```
    COUNT word unsigned;
```

```
    SIZE word unsigned;
```

```
    TYPE byte unsigned;
```

```
    SUBTYPE byte unsigned;
```

```
    constant FIXED_LEN equals . prefix VCLS tag K;
```

```
    constant FIXED_LEN equals . prefix VCLS tag C;
```

```
/* JMT back pointer
```

```
/* number of VLEs in VCL
```

```
/* number of VCB addresses in VCL
```

```
/* size of structure
```

```
/* type of data structure
```

```
/* subtype of data structure
```

```
end VCLDEF;
```

```
end_module $VCLDEF;
```

```
module $VLEDEF;
/*+
/* VLE - VCB List element
/*
/* The VCL contains the VCB addresses of VCBs of journals that have been
/* created for a given tape group. The JMT of the head-JMT for that group
/* points to this VCL. The VCL contains VLEs, each of which, when in use,
/* points to a VCB.
/*
/*-

aggregate VLEDEF structure fill prefix VLE$;
  STATUS_OVERLAY union fill;
    STATUS word unsigned;          /* status
    STATUS BITS structure fill;
      PURGED bitfield mask;
    end STATUS BITS;
  end STATUS_OVERLAY;
  FILL 1 word fill prefix VLEDEF tag $$;          /* spare
  VCB longword unsigned;          /* VCB address
  constant "LENGTH" equals . prefix VLE$ tag K;
  constant "LENGTH" equals . prefix VLE$ tag C;
end VLEDEF;

end_module $VLEDEF;
```


0045 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

INLBUFR
R32

INLDEFINT
SDL

CJFV4

INLPREFIX
R32

CJFUFMAC
SDL

RUFUSR
SDL

INLFILE
SDL

UPGRADE
LIS

BOPTIONS
R32

INLDEF
SDL